Notes

Do FixCapitalization and Plagiarism again, in Lab05 directory, by Friday Sept 23

- Discussed FixCapitalization last time (see video)
- Will discuss Plagiarism today

MyCardDeck (Lab06) now due Monday Sept 26

- Will discuss today

Midterm Oct 5 See CourseInfo/README for topics

Please fill out ANONYMOUS survey on how to improve course. LINK is in CourseInfo/README. By Thurs nite please

(For those who want it…) OPTIONAL hopefully fun project to extend MyCardDeck. LINK is in CourseInfo/README

Last Time

Wrapper classes: Byte, Short, Integer, Long, Float, Double, Character, Boolean

IMMUTABLE classes

- Wrapper classes
- String

```
String lunch = "What's for lunch?";
void LunchAnswer(String x) { x += " Leftovers!"; }
LunchAnswer(lunch);
lunch.toUpperCase();
System.out.println(lunch);
```

'lunch' value DOES NOT CHANGE

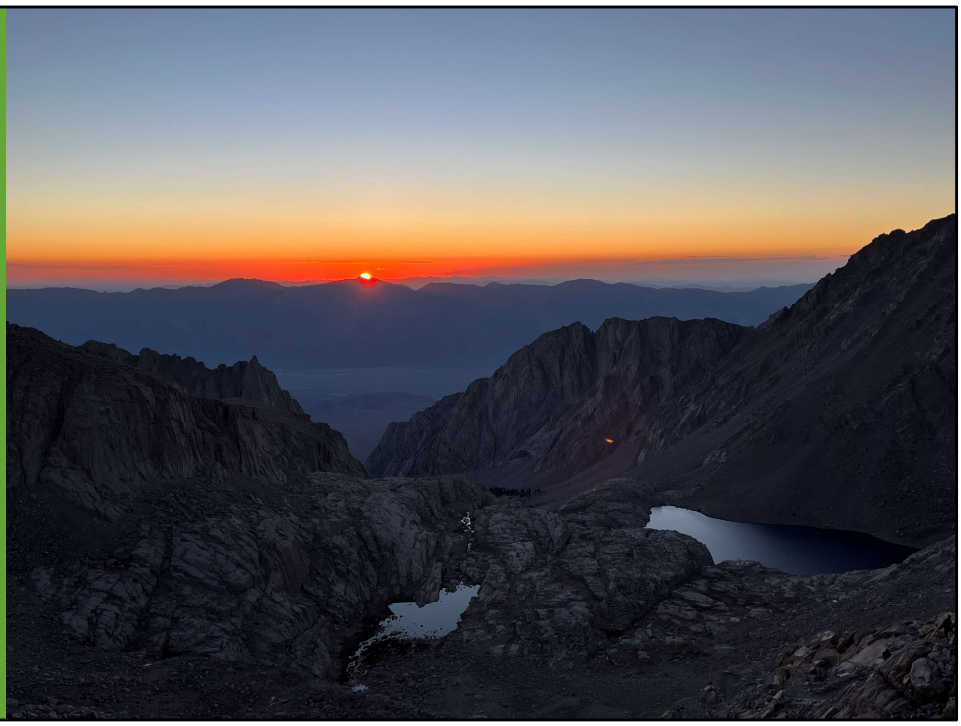
SW Coding Session

Plagiarism.java

MyCardDeck.java

Video shows SW development for BOTH programs

Method Overloading again



Method Overloading

Ok

```
class Student {
      void set(String name);
      void set(int idNumber);
      void set(Boolean hasGraduated);
```

How about?

```
      int get(); // return idNumber
      String get(); // return name
      boolean get(); // return graduated
```

ONLY CAN OVERLOAD BY ARGUMENT, NOT RETURN TYPE
Too much automatic conversion if based on return types:
boolean status(); long status(); Sys.out.println(status());

Method Overloading

How about this?

```
class Difficult {
     void fcn(int a, double b) { System.out.println("int, dbl"); }
     void fcn(double a, int b) { System.out.println("dbl, int"); }
}
…
     Difficult dd = new Difficult();
     dd.fcn(2, 3);
```

Not allowed! "Ambiguous method"

One more thing!

Can use a function call anywhere that its return type is needed

```
class Company {
  int status() { return statusVariable; }
  void printInt(int value) { System.out… }

  void monthlyReport(…) {
       printInt( status() );
  }
}
```

Variable Scope

Scope of a variable

"Scope" of a variable is where in your source code the variable can be used.

When a variable is defined in a class method, it can be used anywhere inside the method (below where it is defined)

- But not outside the method

```
class myClass {
 void myFunction() {
  int x = 2;
  System.out.println("value is " + x);
 }
 void otherFunction() { System.out.println(x); }
}
```

GREEN is ok
RED is ERROR

Scope of a variable

When a variable is defined in a "code block", it can be used anywhere inside the code block.

```
class myClass {
 void myFunction() {
  if (time == 0) {
    int x = 2;
    ...
  }
  System.out.println("value is " + x);
 }
```

ERROR

Scope of a variable

When a variable is defined at the "class level" (a "class instance variable"), it can be used:

- anywhere inside the class, directly (in any member function)
- in other classes, accessed through an object of the class (if "accessible")

ACCESSIBLE = public (or in same package and not private)

Scope of a variable

```
class A {
     public int memberOfA = 5;
     private double privateMember = memberOfA;

     String toString() { return new String(memberOfA); }
}

class B {
 int bVal;

  B() {
     A myAObj = new A();
     bVal = 10 + myAObj.memberOfA;
   }
  int problem() { return 20 + myAObj.memberOfA; }
}
```

`this` keyword

`this` lets source code refer to the current object.

We can access a "class level" instance variable even if hidden by a "method variable".

```
class myClass {
 int x = 1;
 void myFunction() {
    System.out.println("x is " + x);
    int x = 2;
    System.out.println("x is " + x);
    System.out.println("this.x is " + this.x);
 }
```

PRINTS 1 <cr> 2 <cr> 1

Scope of a variable

- How about this?

```
class myClass {
 void myFunction() {
  int x = 1;
  if (time == 0) {
    int x = 2;
  }
  System.out.println("x is " + x);
 }
}
```

Not legal Java. (Yes legal C++)